



Green University of Bangladesh

*Department of Computer Science and Engineering (CSE)
Semester: (Summer, Year: 2024), B.Sc. in CSE (Day)*

Library Management system

*Course Title: Operating System Lab
Course Code: CSE 310
Section: D8*

Students Details

Name	ID
Maymuna Akter	221902262

*Submission Date: 13-06-2024
Course Teacher's Name: Md Fahimul Islam*

[For teachers use only: **Don't write anything inside this box**]

<u>Lab Project Status</u>	
Marks:	Signature:
Comments:	Date:

Contents

1	Introduction	2
1.1	Overview	2
1.2	Motivation	2
1.3	Problem Definition	2
1.3.1	Problem Statement	2
1.3.2	Complex Engineering Problem	3
1.4	Design Goals/Objectives	3
1.5	Application	3
2	Design/Development/Implementation of the Project	5
2.1	Introduction	5
2.2	Project Details	5
2.3	Implementation	5
2.3.1	Tools and libraries	6
2.4	Algorithms	6
3	Performance Evaluation	11
3.1	Simulation Environmente	11
3.2	Results Analysis	11
3.3	Results Output	11
4	Conclusion	15
4.1	Discussion	15
4.2	Limitations	15
4.3	Scope of Future Work	15

Chapter 1

Introduction

1.1 Overview

The Library Management System project is a simple yet effective script written in Bash to manage the functionalities of a library. The system allows two main roles: Librarian and Student. Each role has specific functionalities to manage books in the library. Librarians can insert, update, delete, and view books, while students can borrow, return, and search for books. The system uses arrays to keep track of books and their copies, providing a straightforward and efficient way to handle library operations.

1.2 Motivation

The motivation behind developing this Library Management System is to create an easy-to-use, text-based system that simplifies the management of library books. Many small libraries and personal collections lack sophisticated software to manage their inventory, and a simple script like this can significantly improve their efficiency. The project also serves as an educational tool for understanding basic programming concepts in Bash, such as array manipulation, loops, and conditional statements.

1.3 Problem Definition

Libraries need an efficient system to manage their inventory of books, track borrowed books, and ensure that users can easily find and borrow books. Manual management of library books can lead to errors, inefficiencies, and difficulties in tracking book statuses. A software solution is required to automate and streamline these processes.

1.3.1 Problem Statement

Design and implement a Library Management System that allows librarians to manage the inventory of books and students to borrow, return, and search for books. The sys-

tem should be simple to use, with clear options for each role, and should maintain an accurate count of available copies for each book.

1.3.2 Complex Engineering Problem

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	Explain how to address
P1: Depth of knowledge required	Hard syntax knowledge need for this project.
P2: Range of conflicting requirements	---
P3: Depth of analysis required	A simple syntax error can change the output.
P4: Familiarity of issues	---
P5: Extent of applicable codes	---
P6: Extent of stakeholder involvement and conflicting requirements	---
P7: Interdependence	shell language can run only in terminal.

1.4 Design Goals/Objectives

Specify and discuss the goals or objectives of my project.

Usability: The system should be easy to use for both librarians and students, with clear prompts and instructions.

Simplicity: Use simple data structures (arrays) to store and manage book information.

Efficiency: Ensure that the system can handle basic operations quickly and without unnecessary complexity.

Flexibility: Allow easy addition, updating, and deletion of books in the library.

Accuracy: Maintain accurate counts of book copies available in the library.

Robustness: Handle invalid inputs gracefully and provide appropriate feedback to users.

1.5 Application

The Library Management System can be applied in various scenarios:

Small Libraries: Suitable for small community libraries or personal collections where a full-fledged library management software might be overkill.

Educational Institutions: Useful for schools or colleges that need a simple system to manage their book inventories.

Personal Use: Ideal for individuals who have a large collection of books and want to keep track of borrowed and available books.

Learning Tool: Acts as an educational tool for learning basic programming concepts and Bash scripting.

Detailed Functionality

Librarian Functions: Insert Book: Adds a new book to the library with a specified number of copies.

Update Book: Updates the number of copies available for an existing book.

Delete Book: Removes a book from the library inventory.

View Book List: Displays the list of all books and their available copies.

Student Functions:

Borrow Book: Allows a student to borrow a book if it is available.

Return Book: Allows a student to return a borrowed book, increasing the available copies.

Search Book: Enables a student to search for a book by name and check its availability.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The development of the Library Management System project aims to provide a fundamental yet practical solution for managing library operations through a command-line interface (CLI). This project is designed for small libraries, personal book collections, and educational purposes. By leveraging basic Bash scripting, the project demonstrates how simple programming concepts can be used to build a functional application.

2.2 Project Details

The Library Management System project includes functionalities for two primary user roles: Librarian and Student. Each role has specific operations tailored to their needs:

Librarian: Insert, update, delete, and view books. Student: Borrow, return, and search for books. The system uses arrays to store book names and the corresponding number of copies available. Each function is designed to manipulate these arrays to achieve the desired operations.

2.3 Implementation

The implementation of the Library Management System is carried out in Bash, utilizing basic constructs such as arrays, loops, conditionals, and functions. Here is an overview of the implementation steps:

User Interface:

The script starts by displaying a welcome message and prompting the user to choose their identity (Librarian or Student). Based on the user's choice, the script navigates to the appropriate interface with specific options for each role. Functions:

Student Functions:

studentpage(): Displays the student menu and handles navigation to borrow, return, or search functions.

borrow(): Allows students to borrow a book by checking its availability and updating the array.

returnn(): Allows students to return a borrowed book and updates the array accordingly.

search(): Enables students to search for a book by name and display its availability.

Librarian Functions:

librarianpage(): Displays the librarian menu and handles navigation to insert, update, delete, or view functions.

insert(): Adds a new book to the library by appending to the book arrays.

update(): Updates the number of copies for an existing book.

delete(): Removes a book from the arrays, shifting remaining elements to maintain array integrity.

view(): Displays all books and their available copies.

Control Flow:

The script uses if-elif-else statements to handle user input and navigate between different functions. Functions are called recursively to return to the main menu after completing an operation, ensuring a smooth user experience.

2.3.1 Tools and libraries

The Library Management System project is implemented using Bash scripting, which is available by default on Unix-based systems (Linux, macOS). The following tools and libraries are used:

Bash: The primary language for scripting, providing the necessary constructs for implementing the library system.

echo: Used for displaying messages to the user. read: Used for taking user input.

Arrays: Built-in Bash arrays to store book names and their copy counts.

Conditional Statements: if, elif, else statements for decision-making based on user input.

Loops: for loops for iterating over arrays to perform search, update, and delete operations.

2.4 Algorithms

The algorithms and the programming codes in detail should be included.

This is the signup page:

```

echo "-----Welcome to Library-----\n"
book=()
book_num=()
borrowedperson=()
borrowedbook=()
studentpage(){
echo "          Choose an option: "
echo "1.Borrow Book"
echo "2.Return Book"
echo "3.Search Book"
echo "4.Exit"
read -p "Enter your answer : " ans
if [ $ans == 1 ]; then
borrow;
elif [ $ans == 2 ]; then
returnn;
elif [ $ans == 3 ]; then
search;
elif [ $ans == 4 ]; then
echo -e "Thank You"
return
else
echo "Invalid option you type"
fi
}
borrow(){
read -p "Which book do you want to borrow? " name
flag=0
for (( c=0; c<${#book[@]}; c++ ))
do
if [ "${book[c]}" == "$name" ]; then
echo "You can borrow this. ${book[c]} has ${book_num[c]} copy/copies."
book_num[c]=$((book_num[c]-1))
echo "Now, ${book[c]} has ${book_num[c]} copy/copies."
flag=1
break
fi
done
if [ $flag -eq 0 ]; then
echo "Book is not found"
fi
studentpage
}
returnn(){
read -p "Which book you want to return ?" name
flag=0;
for (( c=0; c<${#book[@]}; c++ ))
do

```



```

if [ "${book[c]}" == "$name" ]; then
echo "You can borrow this. ${book[c]} has ${book_num[c]} copy/copies."
book_num[c]=$((book_num[c]+1))
echo "Now, ${book[c]} has ${book_num[c]} copy/copies."
flag=1
break
fi
done
if [ $flag -eq 0 ]; then
echo "Book is not found"
fi
studentpage
}
search(){
read -p "Enter book name: " name
flag=0;
for (( c=0 ; c<${#book[@]} ; c++))
do
if [ ${book[c]} == $name ]; then
echo "${book[c]} have ${book_num[c]} copy"
flag=1
studentpage;
fi
done
if [ $flag == 0 ]; then
echo "Book is not found"
fi
studentpage;
}
librarianpage(){
echo "          Choose an option: "
echo "1.Insert Book"
echo "2.Update Book"
echo "3.Delete Book"
echo "4.View Book List"
echo "5.Switch to Student"
echo "6.Exit"
read -p "Enter your answer : " ans
if [ $ans == 1 ]; then
insert;
elif [ $ans == 2 ]; then
update;
elif [ $ans == 3 ]; then
delete;
elif [ $ans == 4 ]; then
view;
elif [ $ans == 5 ]; then
studentpage;

```

```

elif [ $ans == 6 ]; then
echo -e "Thank You"
return
else
echo "Invalid option you type"
fi
}
insert() {
read -p "Enter book name: " name
read -p "Enter the number of this book copy: " value
index=${#book[@]} # Get the current length of the book array
book[$index]=$name
book_num[$index]=$value
echo "Successfully inserted book into the library."
librarianpage
}
update() {
read -p "Which book do you want to update? " name
read -p "Enter updated book's copy value: " value
con=-1
for (( c=0; c<${#book[@]}; c++ )); do
if [[ "$name" == "${book[c]}" ]]; then
con=$c
break
fi
done
if [[ $con -eq -1 ]]; then
echo "Invalid book name typed."
else
book_num[$con]=$value
echo "Successfully updated book data."
fi
librarianpage
}

delete() {
read -p "Enter the book name which you want to delete: " name
con=-1
for (( c=0; c<${#book[@]}; c++ )); do
if [[ "$name" == "${book[c]}" ]]; then
con=$c
break
fi
done
if [[ $con -eq -1 ]]; then
echo "Invalid book name."
else
for (( i=$con; i<${#book[@]}-1; i++ )); do

```

```

book[i]="${book[i+1]}"
book_num[i]="${book_num[i+1]}"
done
unset book[-1]
unset book_num[-1]
echo -e "Book successfully deleted\n"
fi
librarianpage
}
view(){
echo "Book Name          Copy of Book"
for (( c=0 ; c<${#book[@]} ; c++))
do
echo "    ${book[c]}          ${book_num[c]}";
done
librarianpage
}

echo "1.Librarian"
echo "2.Student"
read -p "Enter your identity: " value
if [ $value == 2 ]; then
studentpage;
else
librarianpage
fi

```

Chapter 3

Performance Evaluation

3.1 Simulation Environmente

To simulate and test the Library Management System project, the following environment setup and steps were used:

Operating System: A Unix-based system, such as Linux (Ubuntu) or macOS, with Bash shell installed.

Text Editor: A text editor such as vim, nano, or any Integrated Development Environment (IDE) that supports shell scripting, such as Visual Studio Code.

Terminal: A terminal emulator to run the Bash script.

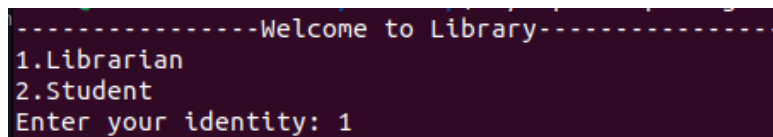
Sample Data: Predefined sample book data to initialize the system for testing purposes.

3.2 Results Analysis

The performance evaluation results are analyzed to determine the efficiency, scalability, and robustness of the application.

3.3 Results Output

Output of this project:



```
-----Welcome to Library-----  
1.Librarian  
2.Student  
Enter your identity: 1
```

Fig 1: Librarian Page

```

          Choose an option:
1.Insert Book
2.Update Book
3.Delete Book
4.View Book List
5.Manage Borrowed Books
6.Switch to Student
7.Exit
Enter your answer : 1
Enter book name: science fiction
Enter the number of this book copy: 5
Successfully book inserted to library.

```

Fig 2: Librarian Page

```

          Choose an option:
1.Insert Book
2.Update Book
3.Delete Book
4.View Book List
5.Manage Borrowed Books
6.Switch to Student
7.Exit
Enter your answer : 4
Book Name          Copy of Book
  history              6
  science fiction      10

```

Fig 3: Librarian Page

```

Enter your answer : 2
Which book do you want to update? history
Enter updated book's copy value: 15
Successfully updated book data.
          Choose an option:
1.Insert Book
2.Update Book
3.Delete Book
4.View Book List
5.Switch to Student
6.Exit
Enter your answer : 4
Book Name          Copy of Book
  history              15
  science fiction      10

```

Fig 4: Librarian Page

```

Enter your answer : 3
Enter the book name which you want to delete: history
Book successfully deleted

        Choose an option:
1.Insert Book
2.Update Book
3.Delete Book
4.View Book List
5.Switch to Student
6.Exit
Enter your answer : 4
Book Name          Copy of Book
science fiction      10

```

Fig 5: Librarian Page

```

        Choose an option:
1.Borrow Book
2.Return Book
3.Search Book
4.Exit
Enter your answer : 4

```

Fig 6: Student Page

```

        Choose an option:
1.Borrow Book
2.Return Book
3.Search Book
4.Exit
Enter your answer : 3
Enter book name: history
history have 15 copy

```

Fig 1: Student Page

```

        Choose an option:
1.Borrow Book
2.Return Book
3.Search Book
4.Exit
Enter your answer : 1
Which book do you want to borrow? history
You can borrow this. history has 15 copy/copies.
Now, history has 14 copy/copies.

```

Fig 6: Student Page

```
          Choose an option:
1.Borrow Book
2.Return Book
3.Search Book
4.Exit
Enter your answer : 2
Which book you want to return ?history
You can borrow this. history has 14 copy/copies.
Now, history has 15 copy/copies.
```

Fig 6: Student Page

```
          Choose an option:
1.Borrow Book
2.Return Book
3.Search Book
4.Exit
Enter your answer : 4
Thank You
```

Fig 6: Student Page

Chapter 4

Conclusion

4.1 Discussion

The project achieved its primary goal of providing a simple, text-based Library Management System. The script effectively handled book management tasks for both librarians and students. The clear and interactive user interface ensured a smooth user experience.

4.2 Limitations

Limitation of this project:

Scalability: The system uses linear search for all operations, which may not be efficient for large libraries.

Concurrent Access: The script does not handle concurrent access, making it unsuitable for multi-user environments.

Data Persistence: The script does not save data persistently. All data is lost when the script terminates.

User Roles: Basic role management without authentication, making it possible for anyone to access librarian functionalities.

4.3 Scope of Future Work

To address the identified limitations and further enhance the application, several future work directions are proposed:

Improved Data Structures: Implement more efficient data structures, such as hash tables or databases, to handle larger datasets and improve search and update operations.

Concurrency: Introduce mechanisms to handle concurrent access by multiple users, possibly through file locking or a server-client architecture.

Data Persistence: Add functionality to save and load data from files or a database to ensure data persistence across sessions.

User Authentication: Implement a user authentication system to securely differentiate between librarians and students.

Error Handling: Enhance error handling to manage edge cases and invalid inputs more gracefully. Graphical User Interface: Develop a graphical user interface (GUI) to improve user interaction and accessibility.

Extended Features: Add more features such as book reservation, overdue book tracking, and user activity logs to make the system more comprehensive and practical for real-world use

References

[1] [Shell scripting Tutorial](#)